

Contents

1 Routine/Function Prologues	2
1.0.1 readcard.F90 (Source File: readcard.F90)	2
1.0.2 check_timestep (Source File: readcard.F90)	4
1.0.3 openfile (Source File: readcard.F90)	5

1 Routine/Function Prologues

1.0.1 readcard.F90 (Source File: readcard.F90)

Reads in LIS run specifics from lis.crd

REVISION HISTORY:

REVISION HISTORY:

15 Oct 1999: Paul Houser; Initial code
 4 Apr 2000: Jeffrey Walker; Added catchment model output interval
 11 Apr 2000: Brian Cosgrove; Added Elevation correction and Forcing
 Mask read statements
 6 Jun 2000: Jon Radakovich; Updated for new version of CLM
 23 Feb 2001: Urszula Jambor; Added GEOS or GDAS forcing option
 27 Mar 2001: Jon Gottschalck; Revision of subroutine by implementing namelists
 05 Sep 2001: Brian Cosgrove; Altered forcing logfile output to include
 more precip types
 04 Feb 2002: Jon Gottschalck; Added section to set to Koster tilespace files if necessary
 15 Apr 2002: Urszula Jambor; Added ECMWF forcing options, also
 adding 1 & 1/2 degree GLDAS domain options.
 28 Apr 2002: Kristi Arsenault; Added NOAH LSM code
 17 Oct 2003: Yudong Tian; added domain 7: reginal 5km
 14 Nov 2003: Sujay Kumar; Modified card file that includes regional
 modeling options
 17 Nov 2003: Yudong Tian; added domain 8: reginal 1km

INTERFACE:

subroutine readcard

USES:

```
use lisdrv_module, only : lis
use time_module, only : date2time
implicit none
```

CONTENTS:

```
open(10,file='lis.crd',form='formatted',status='old')
!-----
! Reading in parameters that need to be initialized
! to avoid any problems later
! Reading in parameters that are used by all LDAS runs
!-----
read(unit=10,nml=driver)
read(unit=10,nml=lis_run_inputs)

!YDT: get command line arguments and convert to IC, IR
lis%d%IC = 1
lis%d%IR = 1
```

```

ttmp=''
Do i=1, iargc()
  call getarg(i, stmp)
  if (trim(stmp).eq.'-ic') then
    call getarg(i+1, stmp)
    Read(Unit=stmp, *) lis%d%IC
    ttmp=trim(stmp)//'-''
  end if
  if (trim(stmp).eq.'-ir') then
    call getarg(i+1, stmp)
    Read(Unit=stmp, *) lis%d%IR
    ttmp=trim(ttmp)//trim(stmp)
  end if
End Do

if ( lis%o%odirn == 1 ) then
  lis%o%odir_array(1) = trim(lis%o%odir_array(1))/trim(ttmp)
endif
lis%o%odir = lis%o%odir_array(1)
!-----
! Open runtime diagnostics file
!-----
call openfile(name,lis%o%odir,lis%o%expcode,lis%o%ofile)
88 format(a4,25x,a3,5x,16a)
89 format(20x,a49)
!-----
! Impose limit on time step
!-----
call check_timestep(lis%t%ts)
!-----
! Set Time
!-----
call set_time(lis%t)

call set_output_counters(lis%o%numoutf)

call date2time(lis%t%time,lis%t%doy,lis%t%gmt, &
  lis%t%yr,lis%t%mo,lis%t%da,lis%t%hr,lis%t%mn,lis%t%ss)

print*
print*, '***** GSFC-LIS driver *****'
print*, 'experiment code: ',',',lis%o%expcode,','
print*, 'starting time: ',lis%t%smo,'/',lis%t%sda,'/',lis%t%syr
print*, 'ending time: ',lis%t%emo,'/',lis%t%eda,'/',lis%t%eyr
print*
print*, 'ic: ',lis%d%ic
print*, 'ir: ',lis%d%ir
read(unit=10,nml=parms)

```

```

    print*, 'forcing details:'
!-----
! Setting Satellite LAI variables
!-----
    lis%p%laitime = 0.0
    if(lis%p%lai.eq.2) then
        print*, "Using AVHRR Satellite LAI"
    endif
    if(lis%p%lai.eq.3) then
        print*, "Using MODIS Satellite LAI"
    endif

    lis%f%rstflag = 1
    lis%f%gridchange = 0
!-----
! Initialize Statistics files conditional
!-----
    lis%o%foropen=0
!-----
! Fix MAXT out of bounds problems
!-----
    if(lis%d%maxt.gt.lis%p%nt) lis%d%maxt=lis%p%nt
    if(lis%d%maxt.lt.1      ) lis%d%maxt=1
!-----
! Select which vegetation tile space and mask files
!-----
    print*, 'miscellaneous details:'
    if(lis%p%vclass.eq.1) print*, 'avhrr umd vegetation', lis%p%vclass
    if(lis%p%vclass.eq.2) print*, 'modis umd vegetation', lis%p%vclass
    print*, 'mask file: ', lis%p%myfile
    print*, 'vegetation file: ', lis%p%vfile
    if (lis%d%soil.eq.1) print *, 'original vegetation-based soil scheme'
    if (lis%d%soil.eq.2) print *, 'reynolds soils'
    if (lis%d%soil.eq.1) print *, 'original vegetation-based soil scheme'
    if (lis%d%soil.eq.2) print *, 'reynolds soils'

    close(10)
    return

```

1.0.2 check_timestep (Source File: readcard.F90)

check timestep to make sure it is between 1 and 3600

INTERFACE:

```
subroutine check_timestep(timestep)
```

CONTENTS:

```

if (timestep.gt.3600) then
  print *,'-----',
  print *,'error, user timestep > 3600 seconds!!!'
  print *,'resetting lis%ts to 3600!!!!'
  print *,'-----',
  timestep=3600
endif
if(timestep.lt.1) then
  print*, 'Timestep can not be less than 1 minute, reset to 15 min.'
  timestep=15*60
endif
return

```

1.0.3 openfile (Source File: readcard.F90)

This subroutine puts together a filename

REVISION HISTORY:

4 Jan 2000: Paul Houser; Original Code

INTERFACE:

```

subroutine openfile(name,odir, expcode, ffile)
  implicit none

```

OUTPUT PARAMETERS:

```
  character*80 name
```

INPUT PARAMETERS:

```

  character*80 mkdir
  character*40 odir,ffile
  integer :: expcode

```

CONTENTS:

```

!-----
! Put together filename
!-----
92 format(80a1)
93 format(a80)
94 format(a4,i3,a1)
96 format(a40)
100 format(a9)

```

```

write(unit=temp,fmt='(a40)') odir
read(unit=temp,fmt='(80a1)') (fbase(i),i=1,80)
c=0
do i=1,80
    if(fbase(i).eq.( ' ) .and.c.eq.0)c=i-1
enddo

write(unit=temp,fmt='(a4,i3,a1)')'/EXP',expcode,'/
read(unit=temp,fmt='(80a1)')(fcode(i),i=1,80)
d=0
do i=1,80
    if(fcode(i).eq.( ' ) .and.d.eq.0)d=i-1
enddo
write(unit=temp,fmt='(a40)') ffile
read(unit=temp,fmt='(80a1)')(fname(i),i=1,80)
e=0
do i=1,80
    if(fname(i).eq.( ' ) .and.e.eq.0)e=i-1
enddo

write(unit=temp,fmt='(80a1)')(fbase(i),i=1,c), &
    (fcde(i),i=1,d), (fname(i),i=1,e)
read(unit=temp,fmt='(a80)') name
write(unit=temp,fmt='(a9)')'mkdir -p '
read(unit=temp,fmt='(80a1)')(fmkdir(i),i=1,9)
!-----
! Make the directories for the output files
!-----
write(unit=temp,fmt='(80a1)')(fmkdir(i),i=1,9), &
    (fbase(i),i=1,c),(fcde(i),i=1,d)
read(unit=temp,fmt='(a80)')mkdir
call system(mkdir)
return

```